

**In the Claims:**

1. (Currently amended) A method for handling events in a distributed computing environment, comprising:

receiving, by an event message endpoint on a client platform in the distributed computing environment, indications from one or more client processes registering interest in receiving one or more of a plurality of events generated by a service in the distributed computing environment;

the event message endpoint automatically subscribing to the one or more events with the service in response to said indications registering interest in the one or more events received from the one or more client processes such that the event message endpoint becomes subscribed to the one or more events;

receiving, by the event message endpoint over a network, a message in a markup language sent to the client platform in the distributed computing environment from the service in the distributed computing environment, wherein the message is received at the event message endpoint from the service over the network in the distributed computing environment, and wherein the message includes a markup language representation of one of the one or more events generated by the service to which the event message endpoint is subscribed; and

sending, [[by]] from the event message endpoint, the markup language representation of the event to at least one of the one or more client processes registered with the event message endpoint to receive the event.

2. (Previously presented) The method as recited in claim 1, further comprising, prior to said receiving, by an event message endpoint on a client platform in the

distributed computing environment, indications from one or more client processes on the client platform registering interest in receiving one or more of a plurality of events generated by a service in the distributed computing environment:

obtaining a markup language schema on the client platform, wherein said markup language schema defines a message interface for the plurality of events generated by the service; and

automatically constructing the event message endpoint for the client platform according to the markup language schema, wherein said constructing is performed within a runtime environment of the client platform.

3. (Previously presented) The method as recited in claim 1, further comprising the service sending one or more messages each including a markup language representation of an event to subscribers to the event in response to generation of the event by the service.

4. (Previously presented) The method as recited in claim 1, wherein the markup language message from the service includes an authentication credential for the service, the method further comprising the event message endpoint authenticating the markup language message as being from the service according to the authentication credential for the service.

5. (Previously presented) The method as recited in claim 1, further comprising the event message endpoint verifying type correctness of the markup language message according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

6. (Previously presented) The method as recited in claim 1, wherein the markup language schema defines a plurality of messages including markup language

representations of the plurality of events generated by the service, the method further comprising the event message endpoint verifying correctness of the markup language message from the service according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

7. (Previously presented) The method as recited in claim 2, wherein said constructing said event message endpoint is performed by computer-executable message endpoint construction code on the client platform.

8. (Previously presented) The method as recited in claim 1,

wherein said receiving indications from one or more client processes registering interest in receiving one or more of the plurality of events comprises receiving from each of the one or more client processes an event handler callback method for an event handler of the respective client process; and

wherein said sending the markup language representation of the event to at least one of the one or more client processes registered with the event message endpoint to receive the event comprises:

the event message endpoint calling an event handler callback method of each client process registered with the event message endpoint to receive the event; and

the event message endpoint passing the markup language representation of the event to each called event handler.

9. (Previously presented) The method as recited in claim 1, further comprising:

at least one client process unregistering interest in a first event of the service with the event message endpoint; and

the event message endpoint automatically unsubscribing to the first event with the service in response to said unregistering interest by the at least one client process;

wherein the service is configured to not send messages including markup language representations of the first event to event message endpoints that are unsubscribed to the first event.

10. (Previously presented) The method as recited in claim 2, wherein said obtaining a markup language schema comprises receiving the markup language schema of the service in a service advertisement of the service, wherein the service advertisement is a markup language document that defines each event message generated by the service.

11. (Previously presented) The method as recited in claim 1, wherein the one or more client processes are executing within the client platform.

12. (Original) The method as recited in claim 1, wherein the event is a Java event.

13. (Previously presented) The method as recited in claim 1, wherein said ~~data~~ markup language is eXtensible Markup Language (XML).

14. (Currently amended) A device, comprising:

a processor; and

a memory coupled to said processor, wherein said memory comprises program instructions executable by said processor to implement an event message gate unit configured to:

receive indications from one or more client processes registering interest in receiving one or more of a plurality of events generated by a service in a distributed computing environment;

automatically subscribe to the one or more events with the service in response to said indications registering interest in the one or more events received from the one or more client processes such that the event message gate unit becomes subscribed to the one or more events;

receive over a network a message in a markup language, wherein the message is received sent to at the device event message gate unit in the distributed computing environment from the service in the distributed computing environment, wherein the message includes a markup language representation of one of the one or more events generated by the service to which the event message gate unit is subscribed; and

send the markup language representation of the event from the event message gate unit to at least one of the one or more client processes registered with the event message gate unit to receive the event.

15. (Previously presented) The device as recited in claim 14, wherein, to implement the event message gate unit on the device, the program instructions are executable by said processor to:

obtain a markup language schema, wherein said markup language schema defines a message interface for the plurality of events generated by the service; and

automatically construct the event message gate unit according to the markup language schema within a runtime environment on the device.

16. (Previously presented) The device as recited in claim 14, wherein the event message gate unit is further configured to verify type correctness of the received markup language message according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

17. (Previously presented) The device as recited in claim 14, wherein the markup language schema defines a plurality of messages including markup language representations of the plurality of events generated by the service, and wherein the event message gate unit is further configured to verify correctness of the received markup language message according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

18. (Previously presented) The device as recited in claim 15, wherein, to obtain a markup language schema, the device is configured to receive the markup language schema of the service in a service advertisement of the service, wherein the service advertisement is a markup language document that defines each event message generated by the service.

19. (Previously presented) The device as recited in claim 14, wherein the service is configured to send one or more messages each including a markup language representation of an event to subscribers to the event in response to generation of the event by the service.

20. (Previously presented) The device as recited in claim 14, wherein the markup language message from the service includes an authentication credential for the service, wherein the event message gate unit is further configured to authenticate the markup language message as being from the service according to the authentication credential for the service.

21. (Previously presented) The device as recited in claim 15, wherein, to implement the event message gate unit on the device, the program instructions are further executable by said processor to:

obtain an address for said service within the distributed computing environment;

obtain an authentication credential indicating authorization to access said service;  
and

construct the event message gate unit according to the markup language schema,  
the obtained address for the service, and the obtained authentication  
credential for the service.

22. (Previously presented) The device as recited in claim 14,

wherein, to receive indications from one or more client processes registering  
interest in receiving one or more of the plurality of events, the event  
message gate unit is configured to receive from each of the one or more  
client processes an event handler callback method for an event handler of  
the respective client process; and

wherein, to send the markup language representation of the event to at least one of  
the one or more client processes registered with the event message gate

unit to receive the event, the event message gate unit is further configured to:

call an event handler callback method of each client process registered with the event message gate unit to receive the event; and

pass the markup language representation of the event to each called event handler.

23. (Previously presented) The device as recited in claim 14,

wherein the event message gate unit is further configured to:

receive an indication from at least one client process unregistering interest in a first event of the service with the event message gate unit; and

automatically unsubscribe to the first event with the service in response to said indication from the at least one client process unregistering interest in the first event; and

wherein the service is configured to not send messages including markup language representations of the first event to event message gate units that are unsubscribed to the first event.

24. (Previously presented) The device as recited in claim 14, wherein the one or more client processes are executing on the device.

25. (Original) The device as recited in claim 14, wherein the event is a Java event.

26. (Previously presented) The device as recited in claim 14, wherein said



markup language is eXtensible Markup Language (XML).

27. (Currently amended) A device, comprising:

a processor;

a memory coupled to said processor, wherein said memory comprises program instructions executable by said processor to implement a service process configured to:

generate an event;

generate a message in a markup language, wherein the message includes a markup language representation of the event generated by the service process; and

send the message over a network to one or more event message gate units in a distributed computing environment that have each automatically subscribed to the event with the service process in response to one or more client processes registering interest in the event with the respective event message gate unit;

wherein each of the one or more event message gate units [[are]] is operable to distribute the markup language representation of the event sent in the message from the service process to the one or more client processes registered with the respective event message gate unit to receive the event from the service process, wherein to distribute the markup language representation of the event, each of the one or more event message gate units is operable to send the markup language representation of the event from the respective event message gate unit to the one or more client processes.

28. (Original) The device as recited in claim 27, wherein the device further comprises a service message gate unit, wherein said generating a message and said sending the message are performed by the service message gate unit on behalf of the service process.

29. (Previously presented) The device as recited in claim 27, wherein the service process is further configured to:

provide a markup language schema within the distributed computing environment, wherein said markup language schema defines a message interface for a plurality of events generated by the service process; and

wherein each of the one or more event message gate units is constructed according to the markup language schema within a runtime environment on a respective device within the distributed computing environment prior to said one or more client processes registering interest in the event with the respective event message gate unit.

30. (Previously presented) The device as recited in claim 29, wherein the markup language schema defines a plurality of messages including markup language representations of the plurality of events generated by the service process.

31. (Previously presented) The device as recited in claim 29, wherein the service process is further configured to provide the markup language schema in a service advertisement, wherein the service advertisement is a markup language document that defines each event message generated by the service process.

32. (Previously presented) The device as recited in claim 27, wherein the service process is further configured to send one or more messages each including a

markup language representation of an event to event message gate units subscribed to the event in response to generation of the event by the service process.

33. (Previously presented) The device as recited in claim 27, wherein the service process is further configured to attach an authentication credential for the service process to the markup language message, wherein the event message gate units that receive the markup language message are configured to authenticate the markup language message as being from the service process according to the authentication credential attached to the message.

34. (Original) The device as recited in claim 27, wherein the events are Java events.

35. (Previously presented) The device as recited in claim 27, wherein said markup language is eXtensible Markup Language (XML).

36. (Currently amended) A non-transitory computer readable medium comprising program instructions, wherein the program instructions are computer-executable to implement an event message endpoint on a client platform in a distributed computing environment, wherein the event message endpoint is configured to:

receive indications from one or more client processes registering interest in receiving one or more of a plurality of events generated by a service in the distributed computing environment;

automatically subscribe to the one or more events with the service in response to said indications registering interest in the one or more events received from the one or more client processes such that the event message endpoint becomes subscribed to the one or more events;

receive over a network a message in a markup language sent to the client platform in the distributed computing environment from the service in the distributed computing environment, wherein the message is received at the event message endpoint from the service over the network in the distributed computing environment, and wherein the message includes a markup language representation of one of the one or more events generated by the service to which the event message endpoint is subscribed; and

send the markup language representation of the event from the event message endpoint to at least one of the one or more client processes registered with the event message endpoint to receive the event.

37. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein, to implement the event message endpoint on the client platform, the program instructions are computer-executable to:

obtain a markup language schema, wherein said markup language schema defines a message interface for the plurality of events generated by the service; and

automatically construct the event message endpoint for the client platform according to the markup language schema, wherein said constructing is performed within a runtime environment of the client platform.

38. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein the service is configured to send one or more messages each including a markup language representation of an event to subscribers to the event in response to generation of the event by the service.

39. (Previously presented) The non-transitory computer readable medium as

recited in claim 36, wherein the markup language message from the service includes an authentication credential for the service, wherein the event message endpoint is configured to authenticate the markup language message as being from the service according to ~~using~~ the authentication credential for the service.

40. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein the event message endpoint is configured to verify type correctness of the markup language message according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

41. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein the markup language schema defines a plurality of messages including markup language representations of the plurality of events generated by the service, wherein the event message endpoint is configured to verify correctness of the markup language message from the service according to the markup language schema prior to said sending the markup language representation of the event to the at least one of the one or more client processes.

42. (Previously presented) The non-transitory computer readable medium as recited in claim 37, wherein, to implement the event message endpoint on the client platform, the program instructions are further computer-executable to:

obtain an address for said service within the distributed computing environment;

obtain an authentication credential indicating authorization to access said service;

and

construct the event message endpoint according to the markup language schema, the obtained address for the service, and the obtained authentication credential for the service.

43. (Previously presented) The non-transitory computer readable medium as recited in claim 36,

wherein, to receive indications from one or more client processes registering interest in receiving one or more of the plurality of events, the event message endpoint is configured to receive from each of the one or more client processes an event handler callback method for an event handler of the respective client process;

wherein, to send the markup language representation of the event to at least one of the one or more client processes registered with the event message endpoint to receive the event, the event message endpoint is configured to:

call an event handler callback method of each client process registered with the event message endpoint to receive the event; and

pass the markup language representation of the event to each called event handler.

44. (Previously presented) The non-transitory computer readable medium as recited in claim 36,

wherein the event message endpoint is further configured to:

receive an indication from at least one client process unregistering interest in a first event of the service with the event message endpoint; and

automatically unsubscribe to the first event with the service in response to said indication from the at least one client process unregistering interest in the first event;

wherein the service is configured to not send messages including markup language representations of the first event to event message endpoints that are unsubscribed to the first event.

45. (Previously presented) The non-transitory computer readable medium as recited in claim 37, wherein, to obtain a markup language schema, the program instructions are further computer-executable to receive the markup language schema of the service in a service advertisement of the service, wherein the service advertisement is a markup language document that defines each event message generated by the service.

46. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein the one or more client processes are executing within the client platform.

47. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein the event is a Java event.

48. (Previously presented) The non-transitory computer readable medium as recited in claim 36, wherein said markup language is eXtensible Markup Language (XML).